УДК 681.3

М.К. Буза
Белорусский государственный университет, Беларусь
пр. Независимости, 4, г. Минск, 220030

ПРОГРАММНО-АППАРАТНАЯ ПОДДЕРЖКА УСКОРЕНИЯ ВЫЧИСЛЕНИЙ

M.K. Bouza
Belorussian State University, Belarus
4, Independence av., Minsk, 220030

SOFTWARE AND HARDWARE SUPPORT FOR COMPUTATIONAL ACCELERATION

В статье исследуются возможности современных средств для ускорения вычислений. Среди них: компьютеры, графические процессоры, распараллеливание алгоритмов и сжатие данных. Проанализированы стадии подготовки и выполнения задач на графических процессорах и приведены сравнительные оценки всех этапов выполнения программ на Graphics Processing Unit.Сформулированы требования к задачам для их эффективного исполнения на графических процессорах. Предложена компонентная среда проектирования параллельных программ. Дана модификация алгоритма LZW сжатия данных, позволяющая существенно увеличить коэффициент компрессии.

Ключевые слова: ускорение вычислений, графические процессоры, распараллеливание алгоритмов, компрессия данных

The article investigates the possibilities of modern means to speed up calculations. Among them: computers, graphics processors, parallelization of algorithms, and data compression. The stages of preparation and execution of tasks on graphic processors are analyzed and comparative estimations of all stages of execution of programs on Graphics Processing Unit are given. Formulated task requirements for their efficient implementation on GPUs.. A component environment for designing parallel programs is proposed. A modification of the LZW data compression algorithm is given, which allows to significantly increase the compression ratio.

Key words: acceleration of computations, graphic processors, parallelization of algorithms, data compression

Введение

При решении многих задач в области искусственного интеллекта возникает необходимость, с одной стороны, построения сложных алгоритмов их решения, а с другой – обработки больших объемов данных. Достаточно обозначить проблему моделирования функций человеческого мозга. Модель мозга состоит в построении системы, объясняющей физиологические функции мозга через известные законы математики, физики, химии с учетом современных достижений в нейрофизиологии и нейроанатомии.

Основные результаты исследований теоретиков в этой области подтверждают, что главное свойство мозга определяется топологией сети нейронов и динамикой распространения импульсов в ней, которые могут быть воспроизведены с помощью устройств, созданных человеком.

Познания законов, лежащих в основе функционирования человеческого мозга, до сих пор одна из сложнейших задач, над которой работают представители различных нейронаук. Международной командой ученых построена 3D модель человеческого мозга с шагом в 20 микрон надеясь, что это поможет изучить его функции.

Реально модель – это миллиарды нейронов и еще большее количество связей между ними, что делает задачу необычайно сложной. В связи с этим естественно просматриваются две задачи: повышение эффективности обработки алгоритмов и построение методов ускоренной работы с данными.

Для решения этих задач необходимы быстродействующие вычислители, распараллеливание алгоритмов решения задач и данных. Так как над решением задач работают коллективы ученых из различных стран, то для ускорения обмена данными последние необходимо эффективно сжи-

мать. Ниже рассмотрим некоторые подходы для решения указанных проблем.

Вычислительные системы

Для решения сложных задач, как правило, используются суперкомпьютеры, достигающие сотни петафлопс. Среди них в 50й редакции списка TOP 500 (июнь 2018 г) приводятся суперкомпьютеры Summit (США) с производительностью 122.300/187.659 петафлопс и Sunway TaihulLight (КНР) с производительностью 93.015/125.436 петафлопс (реальная и пиковая) соответственно. Такое быстродействие достигается за счет большого количества ядер и широкого использования графических процессоров.

Рассмотрим проблемы и достоинства использования графических процессоров (GPU – Graphics Processing Unit), которые сегодня применяются как для графического рендеринга, так и для общих вычислений GPGPU (General Purpose Graphics Processing Unit). При исследовании возможностей GPU акцентируем внимание на общие вычисления.

Для выявления узких мест функционирования GPU было выполнено ряд экспериментов с использованием технологии CUDA, являющейся кроссплатформенным программным приложением для операционных систем Linux, MacOCX и Windows.

Технология CUDA выбрана в связи с тем, что она предлагает новый путь вычислений: кластерное моделирование потоков, SIMD-инструкции, произвольный доступ к памяти (scatter или gather), позволяет применять разделяемую память, оптимизацию обмена данными между CPU и GPU, возможность низкоуровневого программирования на ассемблере. Все это позволяет сделать более удобным процесс подготовки данных и сократить время вычислений.

Технология CUDA обеспечивает масштабируемость за счет трех ключевых абстракций: иерархии групп потоков, разделенной памяти и барьерной синхронизации. Программисту следует разделить решаемую задачу на ряд подзадач, которые будут выполняться независимо разными блоками потоков. Разделенная память и синхронизация дают возможность потокам работать вместе. Блоки могут выполняться на любом количестве доступных ядер GPU и в любой последовательности, что обеспечивает масштабируемость. Все потоки имеют доступ к единой глобальной памяти. Каждый блок потоков имеет разделенную память, которая видна всем потокам блока.

Для анализа возможностей видеокарты был подготовлен ряд тестовых программ и проанализированы полученные результаты, некоторые из них приведены ниже.

Тестирование и измерение времени выполнения различных этапов тестовых программ были выполнены на видеокарте NVIDIA GEFORCE 9600GT в следующей конфигурации: частота 1625000 MHz, количество регистров на потоковый процессор 8192, количество потоковых процессоров 64, объем константной памяти 65536 байт, объем разделяемой памяти 16384 байт, максимальная размерность блока (block dimension) 512×512×64.

Для сравнения измеряемых параметров тестовые программы были выполнены на CPU Intel Core 2 Duo с тактовой частотой 1,8 GHz.

Для анализа производительности GPU была протестирована программа умножения и последующего сложения матриц (ABC + DE), включая все накладные расходы. Полученные данные приведены в таблице 1.

Таблица 1. Удельный вес выполнения каждого этапа программы

Memory allocation	546718	1,62%
Copy data from host to device	10042897	29,76%
Execution	155327	0,46%
Copy data from device to host	22239977	65,89%
Clean up	766299	2,27%

Из таблицы видно, что большую часть времени занимает операция копиро-

вания данных из памяти CPU к видеокарте и обратно. Время непосредственной обработки занимает доли процента.

Для достижения существенного ускорения алгоритмы решаемых задач должны хорошо распараллеливаться на сотни потоков; содержать минимальное количество сложных для обработки операций: деление, возведение в отрицательную степень и т.д.; незначительное количество ветвлений и минимальный объем обмена данными.

Заметное ускорение достигается, если одни и те же инструкции применяются для больших массивов данных.

На уменьшение коэффициента ускорения существенно влияют: время ожидания в коммуникациях, ограниченная пропускная способность, неэффективные алгоритмы, значительные задержки при запуске большого количества процессов (планирование).

Безусловно, для решения на GPU необходимо тщательно выбирать задачи. Основные приложения для GPU включают: моделирование в физике, обработка и анализ изображений, финансовые расчеты, динамика газов и жидкостей, криптография, обработка звука (методы сжатия), анализ данных (data mining), моделирование свертываемости белка, квантовая химия, вычислительная математика.

В настоящее время графические процессоры трансформировались из устройств с фиксированным набором функций для обработки трехмерной графики в вычислительную платформу для общих вычислений (GPGPU). По сути, теперь каждый, кому необходимо осуществлять быстрые вычисления, может иметь недорогой суперкомпьютер на рабочем столе. Компания nVidia уже интегрировала язык C++ в свою технологию CUDA, что расширяет круг пользователей GPU. Тем не менее, многое еще необходимо делать вручную, например, создавать потоки и планировать доступ к памяти.

Распараллеливание программ

Так как программа включает алгоритм и данные, то рассмотрим некоторые вопросы их параллельной обработки.

На сегодняшний день разработаны различные средства создания параллельных программ [4,5]. При этом каждая среда проектирования имеет свой набор инструментальных средств. Так, например, для языка C++ разработаны собственные многопоточные кроссплатформенные библиотеки. Среди них: Patterns Library, Boost Thread, Threading Building Blocks и другие.

Несмотря на то, что иногда многопоточные C++ программы менее эффективны, чем созданные с помощью низкоуровневых средств, но зато имеют более высокую мобильность и скорость создания. Для ускорения разработки программ и повышения их качества используют паттерны параллельного программирования. Паттерн представляет собой архитектурную конструкцию, являющуюся образцом решения задачи, который после адаптации может быть преобразован в реальный код.

Ниже идет речь о разработанной нами понятной для разработчиков методике проектирования параллельных программ [3].

В отличие от существующих используются более общие модели многопоточных программ и заметно упрощен процесс проектирования. И что необычайно важно, она может использоваться не только создателями программного обеспечения, но и студентами во время обучения процессу проектирования.

Компонентами среды проектирования являются:

1. Общие паттерны, как каркас программы, реализующий некоторую модель создания и функционирования потоков с реализацией коммуникаций, синхронизации и балансирования нагрузки.
2. Специализированные паттерны, конкретизирующие общие паттерны для определенного класса задач.
3. Потокобезопасные контейнеры: вектор, очередь и стек, работающих, например, с очередью заданий.

4. Мастер каркаса, представляющего собой оконный интерфейс пользователя для настройки паттерна.

Кроме того, имеется база готовых приложений и скрипты для экспериментов.

Предлагаемая методика поддерживает все этапы проектирования параллельных программ: формирование функционально независимых фрагментов, нахождение информационных зависимостей, масштабирование, планирование и анализ результатов.

В помощь пользователю подготовлены скрипты для автоматизации запуска многопоточной программы, получения и сохранения характеристик выполнения в файле в форме: время выполнения – количество потоков и вычисления показателей эффективности. Результаты экспериментов выдаются как в текстовом, так и в графическом виде.

Результаты экспериментов показали снижение трудоемкости и подтвердили факт ускорения разработки параллельного кода при использовании предложенной методики по сравнению с универсальными средами программирования. Так, например, при умножении матриц с использованием моделей с равноправными узлами специализированный паттерн по сравнению с общим позволяет уменьшить время разработки программы примерно в 1,8 раза.

Модификация алгоритма Lempel-Ziv-Welch (LZW)

При коллективной работе над проектом приходится многократно обмениваться промежуточными результатами. Это накладные, порой очень существенные, временные затраты. Для ускорения процесса передачи объекты следует сжимать. Существует сегодня ряд методов, позволяющих выполнить эту опцию.

Ниже предлагается модификация известного алгоритма LZW [4] с целью увеличения коэффициента компрессии. Алгоритм LZW заменяет строки символов некоторыми кодами, во время чего и осуществляется сжатие.

Рассмотрим суть модификации и продемонстрируем результаты ее работы на примерах.

Степень сжатия определяется различными факторами. Алгоритм, в частности, работает эффективно, если в потоке данных встречаются повторяющиеся строки любой структуры. В связи с этим алгоритм позволяет сжимать текст на английском языке и различных копий экрана со степенью 50% и выше. Однако, в случае данных другой природы коэффициент сжатия существенно ниже, а иногда «сжатый» файл может превосходить по своим объемам исходный. Такой результат можем получить при сжатии фотореалистичных изображений со всей цветовой палитрой. Эти результаты инициировали работу по развитию алгоритма.

Большинство изображений используют 24 бита для информации о пикселе (8 бит на каждую компоненту RGB-палитры). Пусть имеем графический файл с 256-ю цветами. Если обозначить через w и h соответственно его ширину и высоту, тогда для хранения файла с такими параметрами изображения понадобится $w*h*3$ байта. Эти байты последовательно будут обрабатываться алгоритмом сжатия. Однако, если количество цветов не превосходит 256, что естественно для большинства приложений, можно проиндексировать три байта одним. Тогда $w*h*3$ байта при кодировке алгоритмом LZW преобразуются в $w*h$, что значительно уменьшает объем сжатого файла. Ясно, что для файлов существенного объема мы получим заметный выигрыш.

Заметим, что данную модификацию алгоритма можно адаптировать как к одноцветным, так и к полноцветным изображениям.

Реализованный модифицированный алгоритм опробован на нескольких проиндексированных полноцветных изображениях. Результаты приведены на рисунке 1 для алгоритма LZW при 4, 8, 16, 32, 64, 128 и 256 цветах соответственно.

Проведено сравнение по среднему коэффициенту сжатия для различных классов цветных изображений (от 4-х до полноцветных).

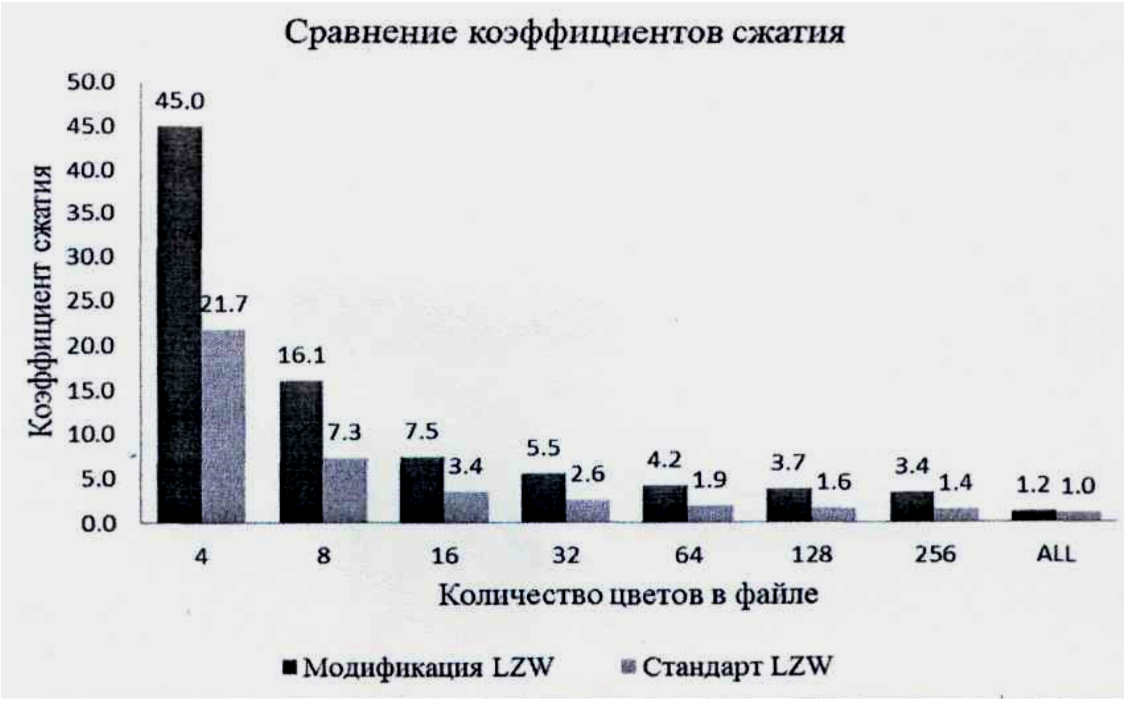


Рисунок 1. Диаграмма сравнений средних коэффициентов сжатия

Как следует из рисунка 1, модифицированный алгоритм позволяет достичь увеличения коэффициента сжатия более, чем в 2 раза.

Заключение

Чтобы достичь желаемого ускорения при обработке сложных задач, необходимо интегрировать и использовать различные инструменты. Среди них: доступные аппаратные средства, в частности, графические, нейро- и квантовые вычислители, суперкомпьютеры. Выбор средств обуславливается спецификой решаемой задачи. Кроме того, по максимуму следует использовать средства распараллеливания алгоритмов и сжатия данных, позволяющие не только ускорить процесс вычислений, но и сократить время коммуникации. Предлагаемые в статье средства в значительной степени позволяют это сделать.

Литература

1. Архитектура CUDA следующего поколения, кодовое название Fermi. Сердце суперкомпьютера в теле GPU.//Сайт компании NVIDIA [Электр. ресурс.]. – 2013. – Режим доступа: http://www.nvidia.ru/object/fermi_architecture_ru.html. – Дата доступа:10.05.2017.
2. Буза, М.К. Архитектура компьютеров // Учебник / М.К. Буза. – Минск: «Вышэйшая школа». 2015. – С. 414.
3. Буза, М.К. Программная среда проектирования параллельных приложений с общей памятью / М.К. Буза, О.М. Кондратьева // Информатика. – 2017. №1. – С.105-111.
4. Профессиональный информационно-аналитический ресурс [электронный ресурс] Все о сжатии данных, изображений и видео. – 2007. – Режим доступа: http: //www. compression.ru – Дата доступа : 04.06.2017.
5. Mattson, T.G. Patterns for Parallel Programming / T.G. Mattson, B.A. Sanders, B.L. Massingill. – Adisson Wesley, 202. – 300 p

References

1. Arkhitektura CUDA sleduyuschego pokoleniya, kodovoe nazvanie Fermi. Serdce supercomp'yutera v tele GPU.//Sayt kompanii NVIDIA [Online]. Available: http://www.nvidia.ru/object/fermi_architecture_ru.html.

2. Bouza, M.K. Architektura komp'yuterov // Uchebnik / M.K. Bouza. – Minsk: «Vysheyshaya shkola». 2015. – P. 414.
3. Bouza M.K. Programmnaya sreda proektirovaniya parallel'nykh prilojeniy s obschey pamyat'yu / M.K. Bouza, O.M. Kondrat'eva // Informatika. – 2017. №1. – C.105-111.
4. Professional'ny'j informacionno-analiticheskiy resurs [Online] Vse o sjatii dannykh, izobrajenij i video. – 2007. – Available: http: //www.compression.ru.
5. Mattson, T.G. Patterns for Parallel Programming / T.G. Mattson, B.A.Sanders, B.L.Massingill. – Adisson Wesley, 202. – 300 p.

RESUME

M.K. Bouza

Software and hardware support for computational acceleration

Today, the problems are formulated, the complexity of which does not allow to implement them on existing computers. Among them: building a model of the human brain, the search for extraterrestrial intelligence, the origin of life on earth, climate prediction and others. In this regard, the task of creating tools to speed up calculations has become urgent.

The performance of modern supercomputers is achieved largely due to the inclusion in their composition of a large number of GPU (Graphics Processing Unit).

The article investigates the different stages of preparation and execution of tasks on the GPU and the specific examples are comparative evaluation of their effectiveness. It is shown that the most time-consuming stage is copy data from device to host. Examples of solving tasks with the use of a computational platform for General purpose computing. Recommendations on the selection of tasks for effective execution on the GPU are given. To speed up the design and solution of complex problems, a parallel program design environment based on General and specialized patterns, secure containers and a window user interface for pattern adjustment is proposed. The method of using the created environment is proposed. It includes the following steps: formation of functionally independent fragments, definetion of information dependencies, scaling, planning and analysis of results. The assessment of the environment efficiency and the areas of application are given.

A modification of the LZW (Lempel-Ziv-Welch) algorithm is proposed to speed up data transmission when working together on the project. It allowed for color images (4 to 256) to achieve increased compression. This number of colors is typical for most applications. The modified LZW algorithm can be adapted to both single-color and full-color images. Integrally proposed hardware and software can significantly speed up the process of solving complex problems by a team of performers.

Надійшла до редакції 05.09.2018